Welcome back hackers!! Today we will be doing a box which is somewhat related to Mirai botnet attack which happened a while ago. The box name is after that attack. So lets get going:

Enumeration

```
PORT      STATE SERVICE REASON         VERSION
22/tcp    open  ssh     syn-ack ttl 63 OpenSSH 6.7p1 Debian 5+deb8u3
(protocol 2.0)
| ssh-hostkey:
|   1024 aa:ef:5c:e0:8e:86:97:82:47:ff:4a:e5:40:18:90:c5 (DSA)
| ssh-dss
AAAAB3NzaC1kc3MAAACBAJpzaaGcmwdVrkG//X5kr6m9em2hEu3SianCnerFwTGHgUHrRpR6iocVh

|   2048 e8:c1:9d:c5:43:ab:fe:61:23:3b:d7:e4:af:9b:74:18 (RSA)
| ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCpSoRAKB+cPR8bChDdajCIpf4p1zHfZyu2xnIkqRAgm6Dws

|   256 b6:a0:78:38:d0:c8:10:94:8b:44:b2:ea:a0:17:42:2b (ECDSA)
| ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBCl89gWp+rA+2SLZzt3r7x+9s

|   256 4d:68:40:f7:20:c4:e5:52:80:7a:44:38:b8:a2:a7:52 (ED25519)
|_ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAILvYtCvO/UREAhODuSsm7liSb9SZ8gLoZtn7P46SIDZL
53/tcp    open  domain  syn-ack ttl 63 dnsmasq 2.76
| dns-nsid:
|_  bind.version: dnsmasq-2.76
80/tcp    open  http    syn-ack ttl 63 lighttpd 1.4.35

|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
| http-methods:
|_  Supported Methods: OPTIONS GET HEAD POST
|_http-server-header: lighttpd/1.4.35
1356/tcp  open  upnp    syn-ack ttl 63 Platinum UPnP 1.0.5.13 (UPnP/1.0
DLNADOC/1.50)
32400/tcp open  http    syn-ack ttl 63 Plex Media Server httpd
```

```
|_http-title: Unauthorized
| http-auth:
| HTTP/1.1 401 Unauthorized\x0D
|_  Server returned status 401 but no WWW-Authenticate header.
|_http-favicon: Plex
|_http-cors: HEAD GET POST PUT DELETE OPTIONS
32469/tcp open  upnp    syn-ack ttl 63 Platinum UPnP 1.0.5.13 (UPnP/1.0
DLNADOC/1.50)
```

There a quite a lot of services which are running on the server, and literally I have no idea about them except SSH, DNS and HTTP. So lets enumerate further what these services are used for and how we can exploit them to gain access.

DNS (Port 53)

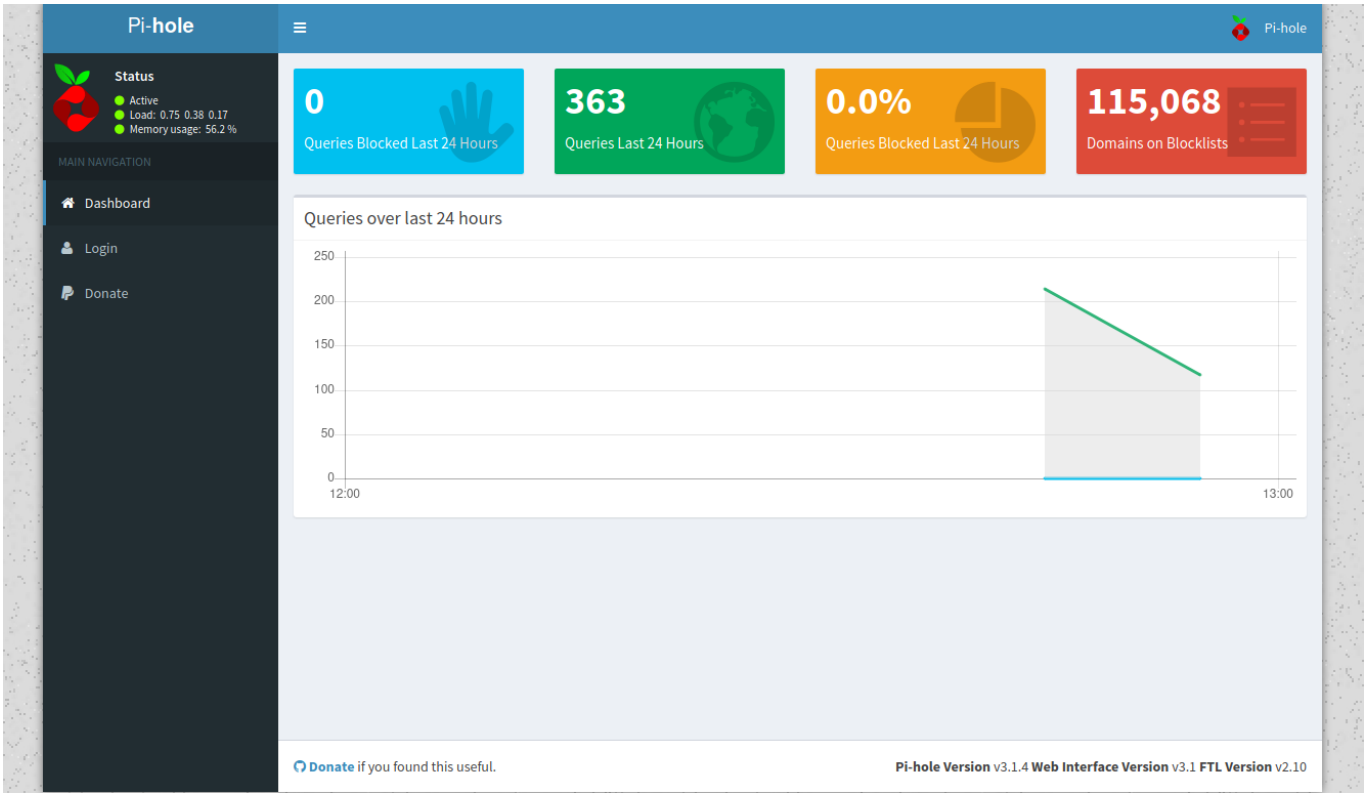
We don't have any information on domain name yet, so zone transfer is out of scope. I also used nsloopkup for reverse lookup but no luck. As we have other ports open too, I will move on further.

Port 80

Nmap reveals that a service with a name of lighttpd is running of version 1.4.35
Lets open our web-browser to explore whats behind. Its just a blank page and wappalyzer lists the web server running behind the server is lighttpd 1.4.35. Next I will run subdirectory bruteforcing using gobuster.

```
┌──(root💀kali)-[/home/rishabh/HTB/Mirai]
└─# gobuster dir -u http://$IP/ -w /usr/share/seclists/Discovery/Web-
Content/directory-list-2.3-medium.txt --no-error -o dirbust -b 400,404 -q
-t 64
/admin                (Status: 301) [Size: 0] [-->
http://10.129.198.228/admin/]
/versions             (Status: 200) [Size: 18]
```

/admin is one of my most favorites subdirectory, when I navigate to this directory, a service called Pi-Hole v3.1.4 is running.



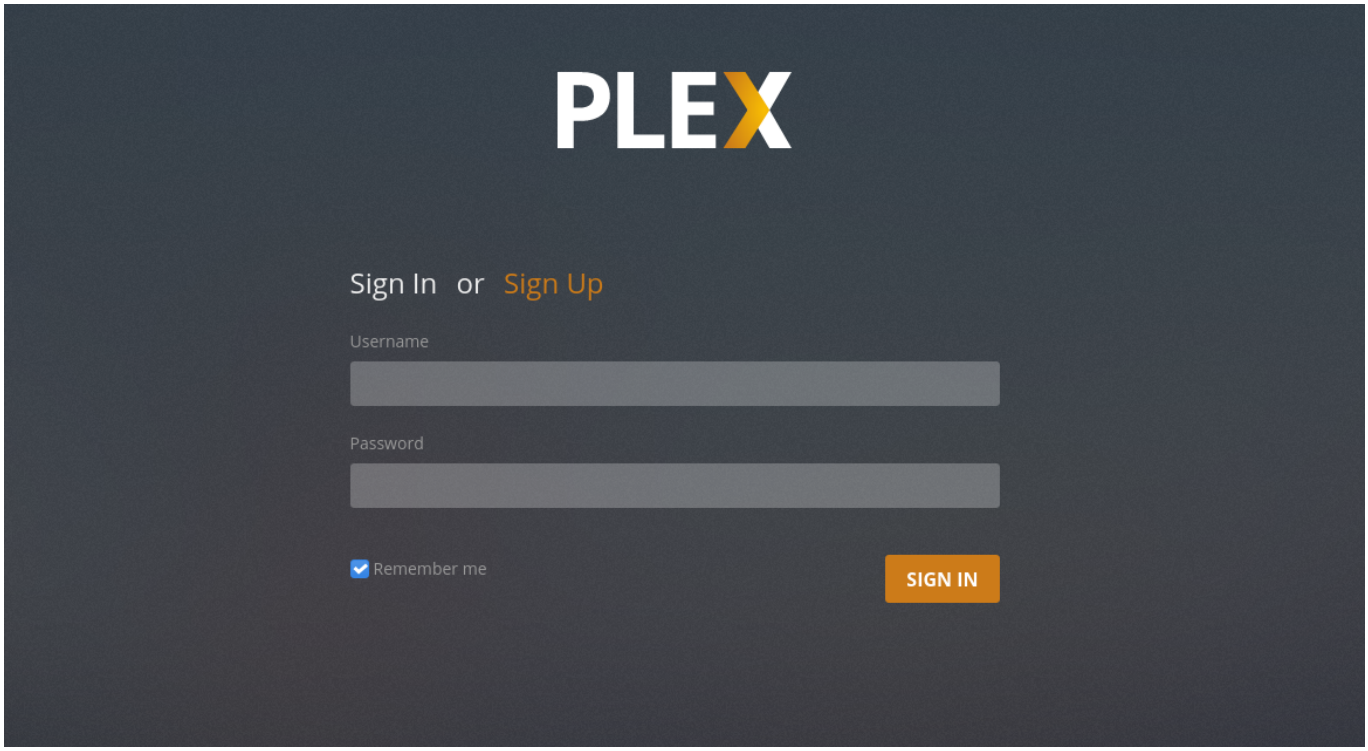
I clicked on login, and googled default credentials, and those were pi:raspberry .I tried some default creds like admin, mirai, raspberry, password but none worked. I also googled exploits for this service, and there is remote code execution for versions < 4.4 but unfortunately it requires authentication and we don't have any creds yet. So I will enumerate other services as well.

Port 1356,32469

I am seeing this service for the first time. I googled exploit for the version name and there was a buffer oveflow associated with it. I tried to interact with the service with both browser and telnet, but it wasn't responding.

Port 32400

Nmap shows its running Plex Media Server. Lets open our browser and see what this final service of the machine has to offer. Home page is just a login page of the service:



I tried default credentials but none worked. Again, google didn't give any specific username and password for this service because it is set just after installation. I ran gobuster on / directory because by default the service redirects to login page.

```
┌──(root💀kali)-[/home/rishabh/HTB/Mirai]
└─# gobuster dir -u http://$IP:32400/ -w
/usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt --
no-error -o dirbust_3 -b 400,404 -q -t 64 --exclude-length 91,101
/security             (Status: 200) [Size: 96]
/web                  (Status: 301) [Size: 0] [-->
http://10.129.198.228:32400/web/index.html]
/identity             (Status: 200) [Size: 175]
/manage               (Status: 301) [Size: 0] [-->
http://10.129.198.228:32400/web/index.html]
```

/identity page reveals possible version number of the Plex service but the version number didn't return any vulnerability.

This XML file does not appear to have any style information associated with it. The document tree is shown below.

<MediaContainer size="0" machineIdentifier="5c27a36a5b4f7104cb9be08579964da1c5de5949" version="1.7.5.4035-313f93718"> </MediaContainer>

To test the workflow, I decided to sign up. I tried admin first but it showed an error saying admin is a valid username. Now comes the trolling, whatever username I was trying to sign up with, it showed an error, username already taken. WOW.

SSH (Port 22)

This version of ssh is vulnerable to username enumeration. I tried some scripts to enumerate some users but none of them gave any positive hits.

Exploitation

Till now, we have default credentials of a pi-hole service which didn't work, and all other services seems to be just rabbit holes. All htb boxes names have a hint to look out for. For example this box's name Mirai refers to the Mirai botnet attack which exploited thousands of IoT devices which were set using default credentials. After gaining access to those devices, Mirai malware got installed and those devices are now being controlled by a remote entity. We do have default credentials at hand but that didn't work for Pi-hole service. I used it to login to Plex service too but no luck. At last I tried SSH and voila I was inside this machine:

```
┌──(root💀kali)-[/home/rishabh/HTB/Mirai]
└─# ssh pi@$IP
The authenticity of host '10.129.198.228 (10.129.198.228)' can't be established.
ED25519 key fingerprint is SHA256:TL7joF/Kz3rDLVFgQ1qkyXTnVQBTYrV44Y2oXyjOa60.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.198.228' (ED25519) to the list of known hosts.
pi@10.129.198.228's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Aug 27 14:47:50 2017 from localhost

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.


SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.

pi@raspberrypi:~ $
```

Privilege Escalation

Sudo -l is my first command when I land on a machine and the output didn't dissappoint me:

```
pi@raspberrypi:/ $ id
uid=1000(pi) gid=1000(pi) groups=1000(pi),4(adm),20(dialout),24(cdrom),27(sudo),29(audio),44(video),46(plugdev),60(games),100(users),101(input),108(netdev),117(i2c),998(gpio),999(spi)
pi@raspberrypi:/ $ sudo -l
Matching Defaults entries for pi on localhost:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User pi may run the following commands on localhost:
    (ALL : ALL) ALL
    (ALL) NOPASSWD: ALL
```

We can run any command without any password using sudo. What else we need. Just type "sudo su -" and you will be root.

```
pi@raspberrypi:/ $ sudo su -

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.


SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.

root@raspberrypi:~# id
uid=0(root) gid=0(root) groups=0(root)
root@raspberrypi:~#
```

Getting to root.txt isin't so straighforward. The usual place of root.txt contained a note to self which said it is in USB stick:

```
root@raspberrypi:~# cat root.txt
I lost my original root.txt! I think I may have a backup on my USB
stick...
```

Now, when I navigate to /media/usbstick, there was another note which had this message:

```
root@raspberrypi:/media/usbstick# cat damnit.txt
Damnit! Sorry man I accidentally deleted your files off the USB stick.
Do you know if there is any way to get them back?

-James
```

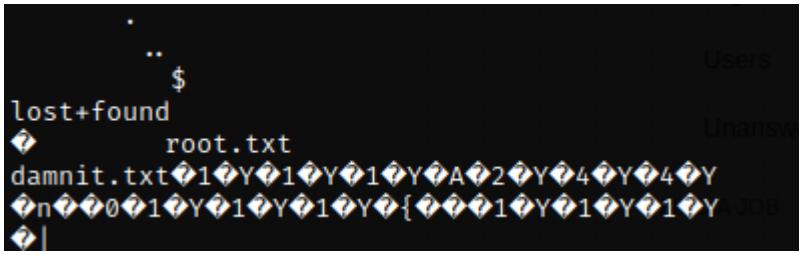
To recover the root.txt, first we need to find the mounted filename of the pendrive. Type fdisk -l and it will list all the attached disks and external drives.

```
root@raspberrypi:/# fdisk -l

Disk /dev/sda: 10 GiB, 10737418240 bytes, 20971520 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x0eddfb88

Device     Boot   Start      End  Sectors  Size Id Type
/dev/sda1  *         64  2709119  2709056  1.3G 17 Hidden HPFS/NTFS
/dev/sda2       2709504 20971519 18262016  8.7G 83 Linux

Disk /dev/sdb: 10 MiB, 10485760 bytes, 20480 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk /dev/loop0: 1.2 GiB, 1297825792 bytes, 2534816 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
root@raspberrypi:/#
```

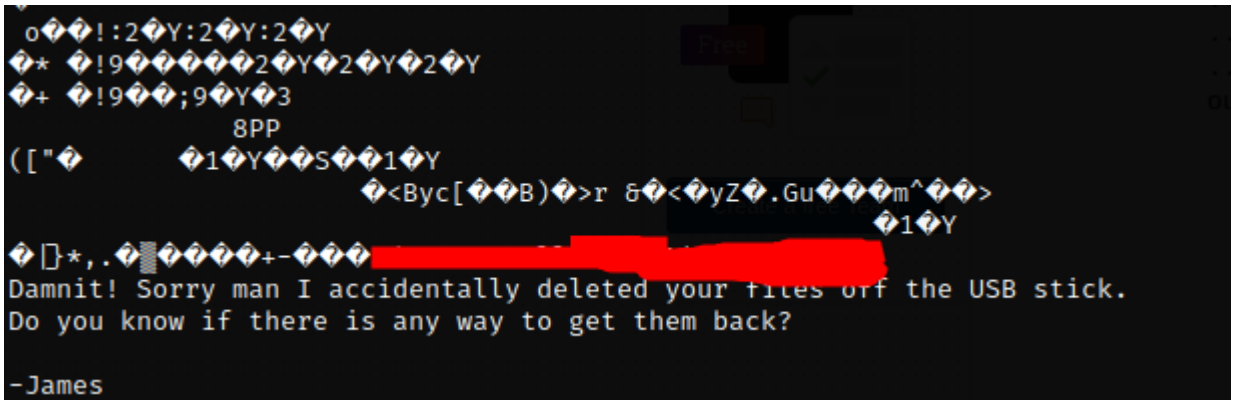
/dev/sda is the host filesystem and /dev/loop0 treats filesystem image as if they were block devices. Now the remaining /dev/sdb which is of 10mb is our mounted pendrive. As you know in Linux, everything is considered a file, you can read the block drive sdb which is the pendrive. So by reading the contents of this file, we can probably read all the contents on the drive which were stored in plaintext. Much of it will be in gibberish but some data will be readable.
If you cat out sdb file in /dev directory:

```
root@raspberrypi:/dev# cat sdb

(Z"@      ;@@a;@@a@@S@@1@Y
                  @<Byc[@@B)@>r &@</media/usbstick@yZ@.Gu@@@m^@@>
                                                        @1@Y
@|*,.@@@@}+-@@n

         !"#$%&'()
```

You can see from the screenshot, we are reading the right file.
Scrolling down you will notice, in the pendrive there consisted 2 files: root.txt and damnit.txt



At the bottom, you will find the root.txt and contents of damnit.txt



Cheers!!